NAME

dgsh-tee − buffer, copy, permute, or distribute data from input sources to output sinks

SYNOPSIS

dgsh-tee [**−b** *buffer-size*] [**−afIMs**] [**−i** *input-file*] [**−o** *output-file*] [**−m** *memory-size*] [**−p** *o1,o2 ...*] [**−T** *directory*] [**−t** *character*]

DESCRIPTION

dgsh-tee will read data from the specified sources and copy or distribute it to the specified sinks. It resembles in its operation *tee*(1) and *cat*(1), but offers additional capabilities required for the operation of *dgsh*(1). In contrast to these programs, *dgsh-tee* will buffer the data it handles, so it will never cause deadlock or starvation when one or more sources are unable to provide data or if sinks are unable to receive them. Furthermore, *dgsh-tee* can copy data from multiple sources to multiple sinks, permute the data between sources and sinks, and also distribute the data among the sinks.

When copying data from a few sources to a multiple of their number sinks, the first input tuple will appear in the first sinks, and so on. As an example, two sources *a, b* will appear in six sinks as *a, b, a, b, a, b*. When copying data from many sources to a fraction of their number sinks, a tuple of sources equal to the number of sinks is output first, followed by a tuple of the next sinks, and so on. As an example, six sources , b, c, d, e, f will appear in two sinks as *a, c, e* in the first sink and *b, d, f* in the second one. In effect, the group of few sources or sinks is treated as a single unit to be scattered or sequentially concatenated.

dgsh-tee is normally executed within *dgsh* through wrappers that replace the system-provided *tee* and *cat* commands. This manual page serves mainly to document its operation, to how it can be used in less common use cases, and to allow the creation of plug-compatible replacements implementing different record types.

OPTIONS

−a Open files subsequently specified with the **-o** option for appending.

−b *buffer-size*
Specify the size of the buffer to use. This is by default 1MB. Buffers are chained together when more space is required, so the main utility of this option is to decrease the buffer size in memory-constrained environments. The specified number can be suffixed with **k**, **M**, *or* **G** *to specify the corresponding unit. The specified buffer size must be less than the program's maximum memory size.*

−f When the allocated memory size reaches the maximum memory threshold, start using a temporary file for buffering the data. This extends the amount of data that can be buffered to the space available on disk. The location of the temporary file follows the *tempnam*(3) rules, and can be overridden through the **-T** option.

−I Implement input-side buffering. By default *dgsh-tee* will buffer only as much input data, as is needed to avoid starving one of the specified sinks. In doing so it may cause its input source to block while having data to write. When this option is enabled *dgsh-tee* will always read data if they are available on the standard input, and will write that data to any (including zero) sinks that can read it. This can be useful in cases where a command with insufficient input buffering, like *join*(1), *sort*(1), or *paste*(1), is gathering input from commands executing in parallel. In such a case adding *dgsh-tee* with input side buffering enabled at the end of each data pipeline, will increase the number of processes that can operate concurrently.

−i *input-file*
Read input from the specified source file, rather than the standard input. The option can be provided multiple times to specify multiple files that will be read sequentially. All input files will be opened at the beginning of the program's operation in order to unblock the execution of

asynchronous shell commands that redirect their output to the corresponding named pipes. Furthermore, when input-side buffering is specified **-I** data is read asynchronously from all specified input files.

−M Provide memory use statistics on termination. This is mainly used for testing, to check against leaks of buffers.

−o *output-file*
Write copies of the input data to the specified sink file, rather than the standard output. The option can be provided multiple times to specify multiple files where input data will be copied.

−m *memory-size*
Specify the maximum size of memory to allocate for buffers. This is by default 256MB. When *dgsh-tee* exhausts this memory, it enters a state where it waits for its output buffers drain, thus freeing allocated memory. The specified number can be suffixed with **k**, **M**, *or* **G** *to specify the corresponding unit. The specified maximum memory size must be larger than the program's buffer size.*

−p *o1,o2 ...*
Permute the inputs to the specified outputs. The comma-separated arguments *o1,o2, ...* specify the number of the output channel (starting from 1) where the corresponding *n*th input channel will be sent. Thus, input 1 goes to output *o1*, input 2 goes to output *o2*, and so on. As an example a cross-permutation is specified with the argument *-p 2,1*.

−s Scatter the input fairly across the sinks, rather than copying it to all. When this option is in effect, the input data are divided into chunks of one or more lines, and each chunk is written only to a single sink. This is useful for dividing the work among multiple processes operating in parallel.

−T *directory*
Specify the directory to use for storing the temporary file, when the specified maximum buffer memory size is exceeded.

−t *char* Use *char* as the record separator, By default the record separator is a newline, An empty (not missing) argument for the record separator will make the record separator be the null character.

SEE ALSO

dgsh(1) *tempnam*(3)

AUTHOR

Diomidis Spinellis — <http://www.spinellis.gr>